

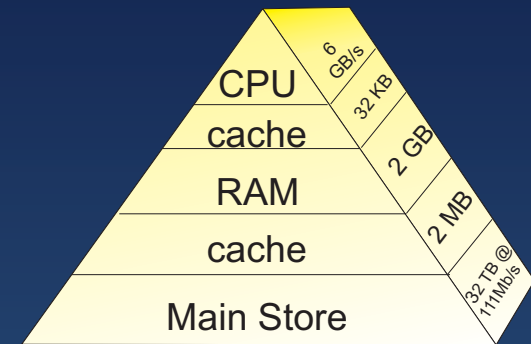
# High Performance Hardware, Memory & CPU

(Step Back, Look Inside)

Rubin H Landau

With

Sally Haerer and Scott Clark

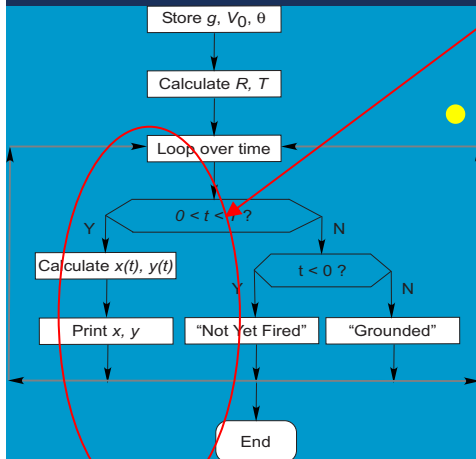


Computational Physics for Undergraduates  
BS Degree Program: Oregon State University

*“Engaging People in Cyber Infrastructure”*  
Support by EPICS/NSF & OSU

# Problem: Optimize for Speedup

- Faster by smarter (algorithm), not bigger Fe
- Yet ... HPC: tune program to architecture
  - 1<sup>st</sup> locate *hot spots*  $\Rightarrow$  speed up?



- Negative side

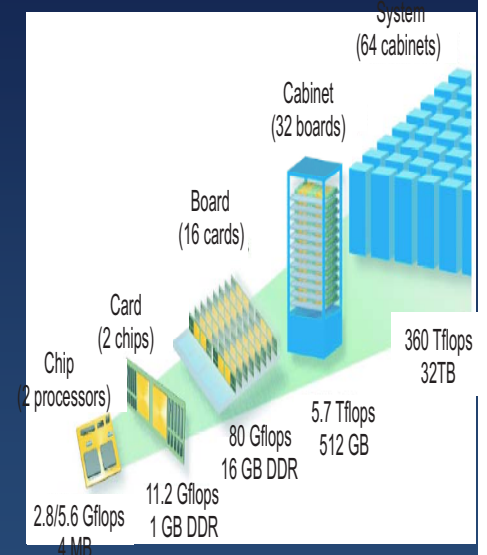
- hard work & (your) time intensive
- local hard/software:  $\downarrow$  portable, readable
- CS: "compiler's job not yours"
- CSE: large, complex, frequent programs: 3-5X
- "CSE: tomorrow's problems, yesterday's HW  
CS  $\leftrightarrow$ " (Press)

# Theory: Rules of Optimization

1. "More computing sins are committed in the name of efficiency (without necessarily achieving it) than for any other single reason - including blind stupidity." - *W.A. Wulf*
2. "We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil." - *D. Knuth*
3. "The best is the enemy of the good." - *Voltaire*
4. Do not do it.
5. (for experts only): Do not do it yet - *M.A. Jackson*
6. Do not optimize as you go. Jonathan Hardwich  
[www.cs.cmu.edu/~jch](http://www.cs.cmu.edu/~jch)
7. Remember the 80/20 rule: 80% results ← 20% effort (also 90/10)
8. Always run "before" and "after" benchmarks.
9. Use the right algorithms and data structures!

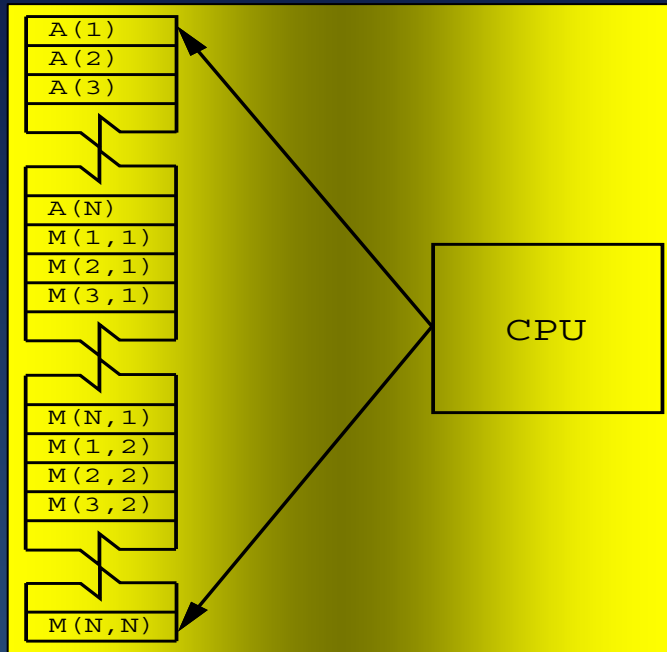
# Theory: HPC Components

- Supercomputers = fastest, most powerful
- Now: parallel machines, PC (WS) based
- Linux/Unix
- HPC = good balance major components:
  - multistaged (pipelined) units
  - multiple CPU (parallel)
  - fast CPU
  - very large, very fast memories
  - very fast communications
  - vector, array processors (?)
  - software: integrates all

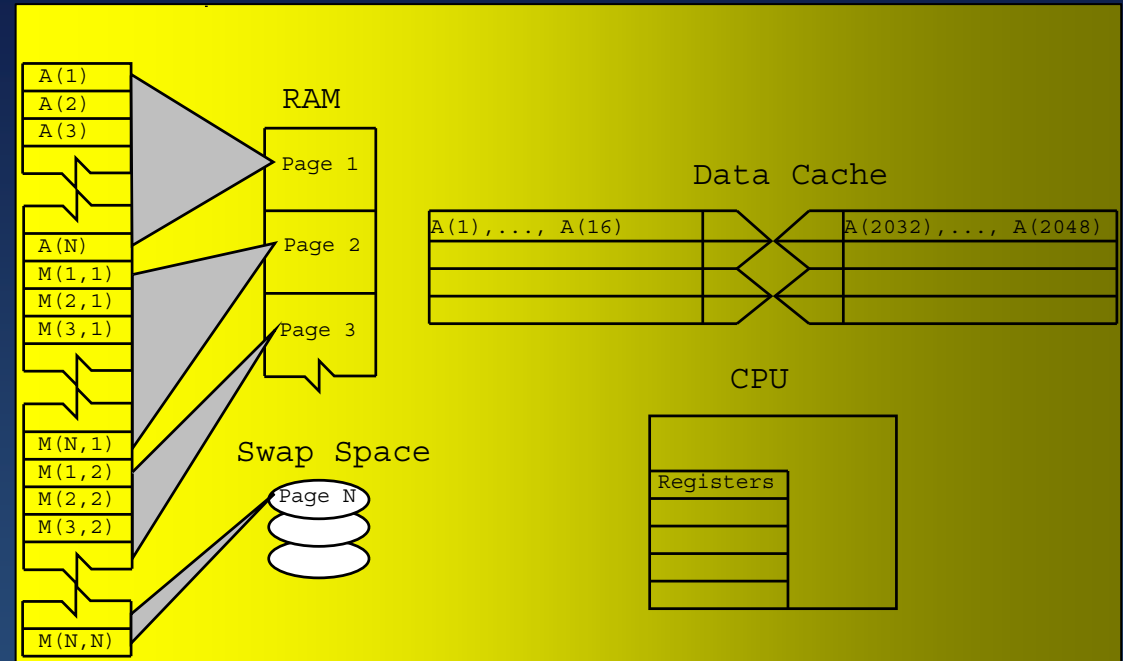


# Memory Hierarchy vs Arrays

Ideal world array storage



Real world matrices  $\neq$  blocks = broken lines



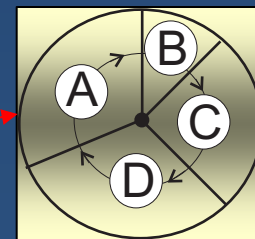
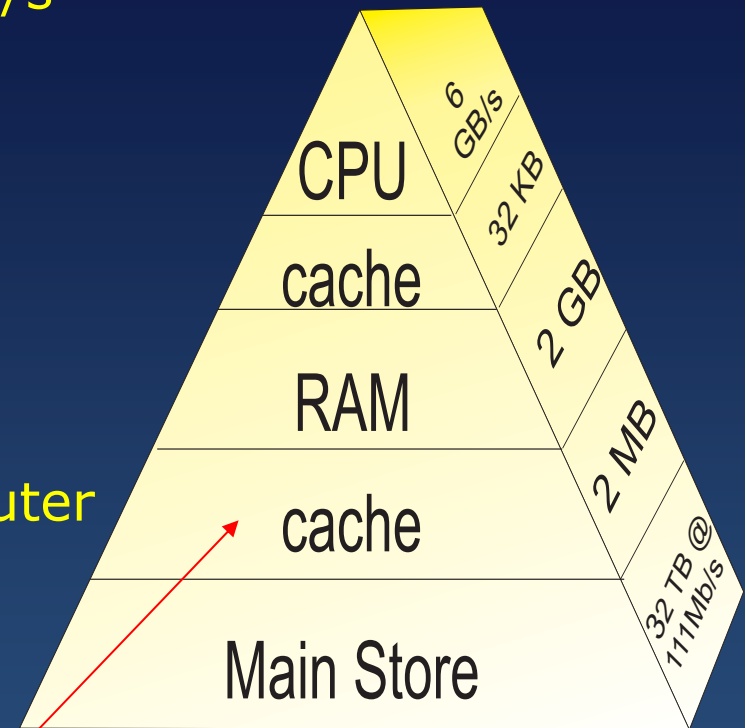
Row major: C, Java;

Column major: F90

- C, J:  $m(0,0)$   $m(0,1)$   $m(0,2)$   $m(1,0)$   $m(1,1)$   $m(1,2)$   $m(2,0)$   $m(2,1)$   $m(2,2)$
- F:  $m(1,1)$   $m(2,1)$   $m(3,1)$   $m(1,2)$   $m(2,2)$   $m(3,2)$   $m(1,3)$   $m(2,3)$   $m(3,3)$

# Memory Hierarchy: Cost vs Speed

- *CPU*: registers, instructions, FPA, 8 GB/s
- *Cache*: high-speed buffer, 5.5 GB/s
- *Cache lines*: latency issues
- *RAM*: random access memory
- *Via RISC*: reduced instruction set computer
- *Hard disk*: cheap and slow, 111 Mb/s
- *Pages*: length = 4K (386), 8-16K (Unix)
- *Virtual memory*  $\approx \infty$  RAM (32b  $\approx$  4GB)  
little effort,  $\uparrow$  \$\$ (t) = *page faults*  
e.g. multitasking/windows



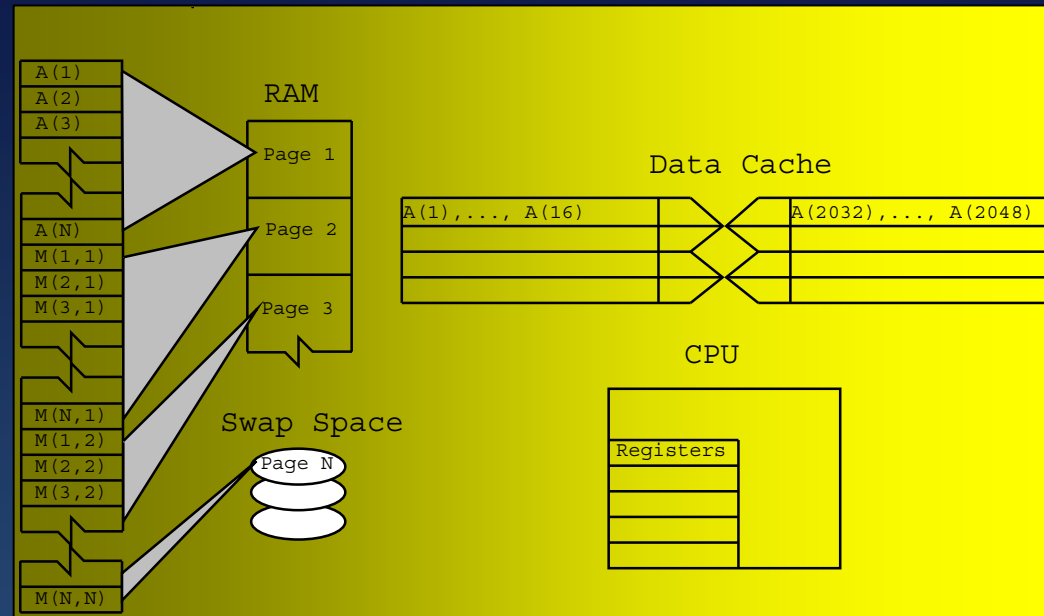
# Central Processing Unit

## Interacting Memories

Pipelines: speed

- Prepare next step during previous
- Bucket brigade

e.g:  $c = (a + b) / (d * f)$



<u>Unit</u>	<u>Step 1</u>	<u>Step 2</u>	<u>Step 3</u>	<u>Step 4</u>
<b>A1</b>	Fetch <i>a</i>	Fetch <i>b</i>	Add	
<b>A2</b>	Fetch <i>d</i>	Fetch <i>f</i>	Multiply	
<b>A3</b>				Divide

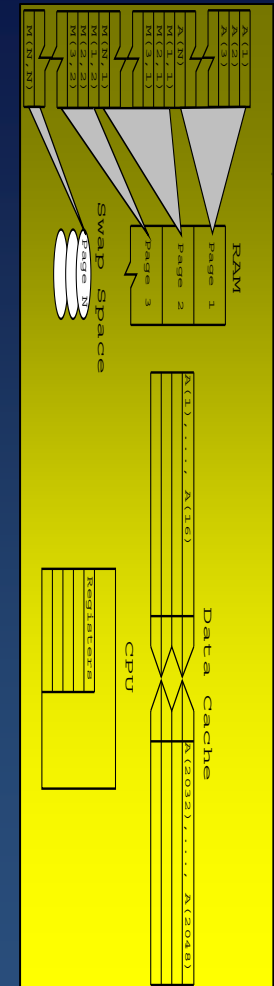
# CPU Design: RISC

- *RISC* = **R**educed **I**nstruction **S**et **C**omputer (HPC)
- $\neq$  *CISC* = **C**omplex **I**SC (previous)
  - high-level microcode on chip (1000's instructions)
  - complex instructions  $\Rightarrow$  slow (10  $\tau$ /instruct)
- RISC: smaller (simpler) instruction set on chip
  - F90, C compiler translate for RISC architecture
  - simpler (fewer cycles/i), cheaper, possibly faster
  - saved instruction space  $\rightarrow$  *more CPU registers*
  - $\uparrow$  pipelines,  $\downarrow$  memory conflict, some parallel
- **Theory**

$$CPU\ T = \# \text{ instructs} \times \text{cycles/instruct} \times \text{cycle } t$$

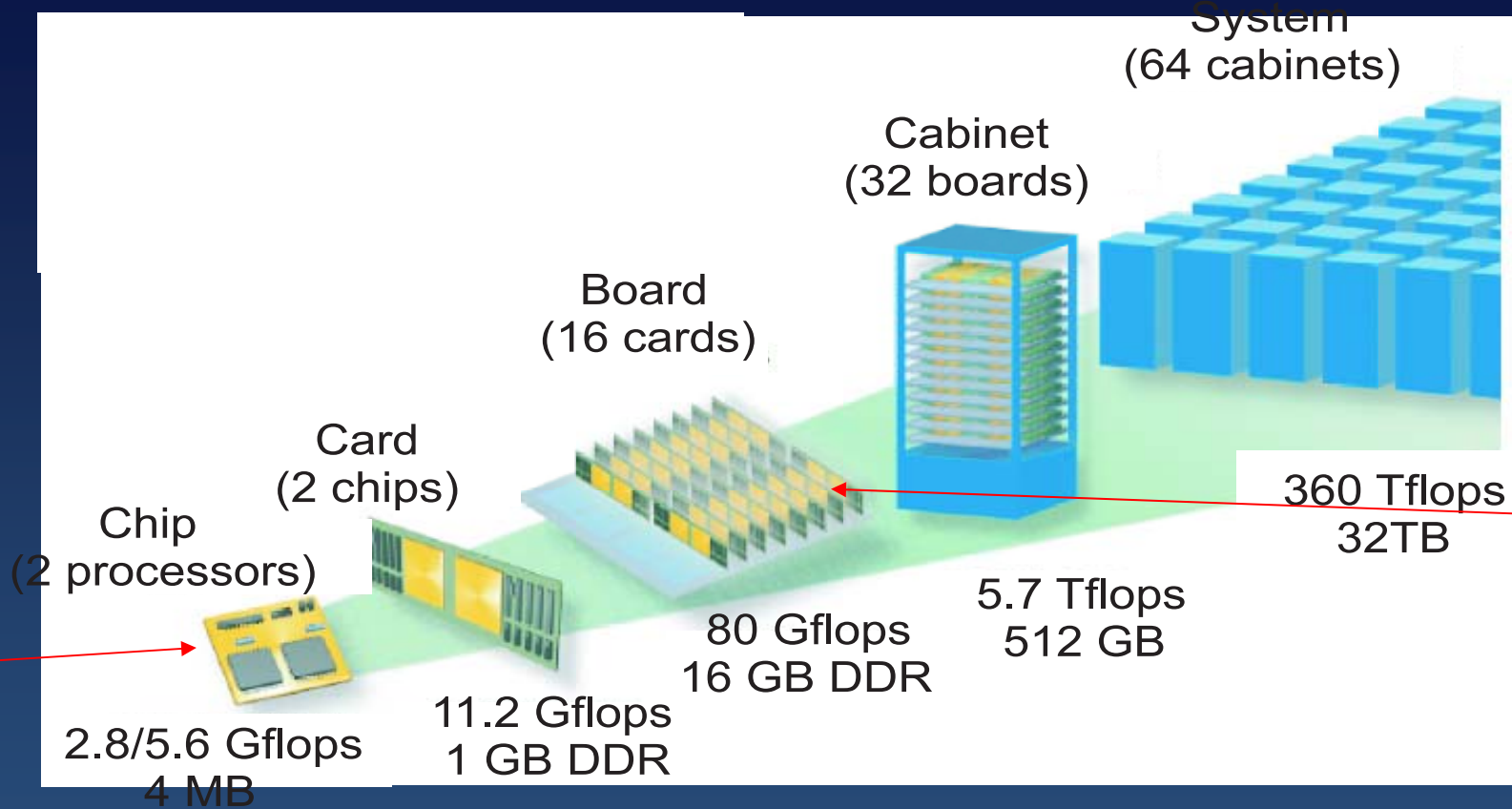
CISC: fewer instructs executed

RISC: fewer cycles/instruct





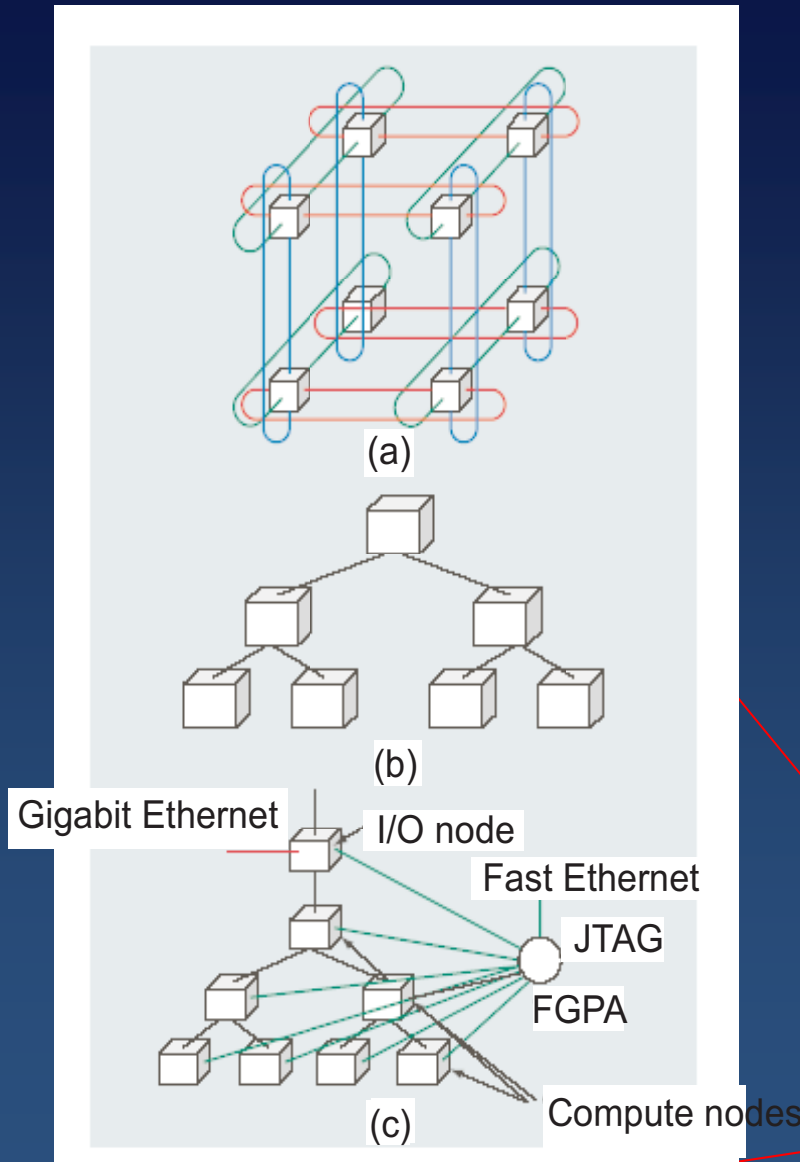
# Latest & Greatest: IBM Blue Gene



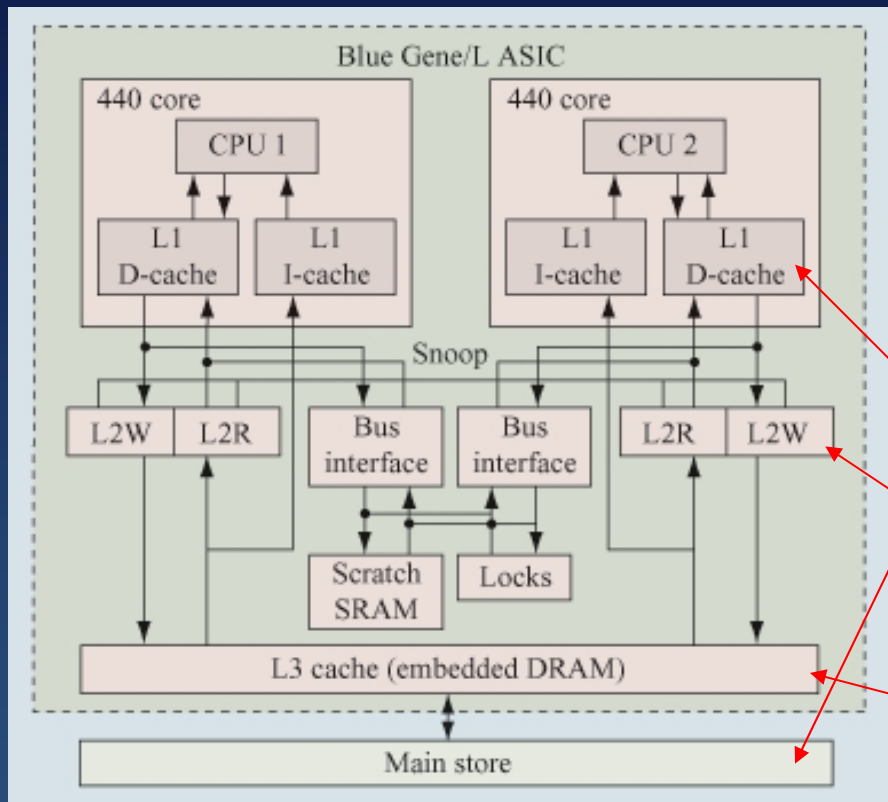
- Balance  $\downarrow$  cost/performance  $\Leftrightarrow$   $\uparrow$  performance/watt
- On, off chip distributed memories
- 2 cores: 1 compute, 1 communicate
- Extreme scale  $\rightarrow$  65,536 ( $2^{16}$ ) nodes
- $\Sigma$  Peak = 360 teraflops ( $10^{12}$ );
- Medium speed 5.6 Gflop (cool)
- 512 chips/card, 16 cards/Board
- Control: distributed memory MPI

# BG's 3 Communication Networks

- Fig (a) ♥: 64 x 32 x 32 3-D torus (2 x 2 x 2 shown)  
links = chips that also compute  
both: nearest-neighbor & cut through  
all  $\approx$  effective bandwidth all nodes  
node  $\Leftrightarrow$  node: 1.4 Gb/s  $\Leftrightarrow$  1 ns
- Program speed: local communication  
100 ns < *Latency* < 6.4  $\mu$ s (64 hops)
- Fig (b) Global collective network  
broadcast to all processors
- Fig (c) Control network + Gb-Ethernet  
for I/O, switch, devices > tb/s



# Blue Gene Compute Heart



- 2 PowerPC 440s, 2 FPU
- 1 Compute, 1 I/O (Ether)
- RISC 7 stage CPU, 3 pipelines
- Memory  $\Sigma$  512 MB/node  $\rightarrow$  32 TB
- Variable page size
- Three Cache Levels

L1 cache: 32 KB

L2 cache: 2 KB

L3 cache: 4 MB

L1: 32 B line width

L2/L3: 128 B

# How to Use this?

- ◆ **Next time!**